ORBIT Measurement Library

OML

WINLAB IAB Meeting
May 2004

M. Singh, I. Seskar, M. Ott
Introduction

- Experimenters measure node, network & application performance, i.e. a lot of numbers.
- How to collect these numbers in a distributed environment like ORBIT?
- How to collect them in real-time?
- OML provides:
  - the distributed software infrastructure to collect them in real-time.
  - flexible and dynamic way to change the way measurements are reported.
OML Requirements

- **Application interface:**
  - Simple and easy interface for application writers.
  - Handle threading issues for application writers.

- **Scalability:**
  - Reduce the network traffic from measurement’s data.
  - Reduce processing for collection and transportation.

- **Controllability:**
  - Control collection behavior without code recompilation
  - Change parameters during the experiment.

- **Use to log application control parameters**
OML Architecture

User application

OML interface to user application

OML data filter, id = xx

OML data filter, id = yy

OML data filter, id = zz

OML transport layer

pluggable filters, chosen by the experimenter

Orbit node

Collection server

SQL DB

OML SQL module

OML XDR decoder

Berkeley DB

OML transport layer

XDR Encoded data over multicast channel.
OML Architecture contd..

**Client:**
- Simple API for application writers
- Filters reduce the amount of reportable data
- XDR encoded data over multicast channel

**Collection Server:**
- One instance of Collection Server per experiment
- Berkeley DB used for scalability
- SQL database for persistent for data archiving
- One multicast channel per experiment for logical segregation of data, and scalability
OML Architecture contd.

- OML API examples
  
  - `oml_group-1(rssi, noise)`
  - `oml_group-2(packet_size)`

  where “group-1” & “group-2” are measurement points defined by the application programmer.

- Data Filter are time based or number based
  
  - `time_avg(packet_size, arr_count, time_trigger, result)`
  - `sample_mean(rssi, arr_count, number_trigger, result)`

  Filter parameters, e.g. “time_trigger” can be changed on the fly.

- Filter output encoded into XDR and multicasted
OML configuration

Orbit node configuration

<multicast-channel addr="224.10.10.1" port="7000"/>
<measurements>
  <group id="group-1">
    <metric id="rssi" type="float"/>
    <metric id="noise" type="float"/>
  </group>
  <group id="group-2">
    <metric id="throughput" type="float"/>
  </group>
</measurements>
Example database setup

- Database schema from experiment definition
- Additional fields for correlation of data.

```
mysql> select * from metrics;
+-----+-------+-------+--------+
<table>
<thead>
<tr>
<th>id</th>
<th>metric</th>
<th>units</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>rssi</td>
<td>db</td>
<td>float</td>
</tr>
<tr>
<td>2</td>
<td>noise</td>
<td>db</td>
<td>float</td>
</tr>
<tr>
<td>3</td>
<td>throughput</td>
<td>bps</td>
<td>float</td>
</tr>
<tr>
<td>4</td>
<td>offered_load</td>
<td>bps</td>
<td>float</td>
</tr>
</tbody>
</table>
+-----+-------+-------+--------+
4 rows in set (0.00 sec)

mysql> select * from metrics_values where id=3 and node_id="orbit4";
+-----+-------+----------+--------+------------+-------------+------------+
<table>
<thead>
<tr>
<th>id</th>
<th>value</th>
<th>packet_size</th>
<th>node_id</th>
<th>sequence_no</th>
<th>timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1086464</td>
<td>256</td>
<td>orbit4</td>
<td>4</td>
<td>1084223553</td>
</tr>
<tr>
<td>3</td>
<td>1515520</td>
<td>256</td>
<td>orbit4</td>
<td>10</td>
<td>1084223554</td>
</tr>
<tr>
<td>3</td>
<td>1503232</td>
<td>256</td>
<td>orbit4</td>
<td>16</td>
<td>1084223555</td>
</tr>
</tbody>
</table>
```
OML current status

- **Controllability & features:**
  - Collection server control is Web Enabled
  - Change collection parameters on the fly
  - Provides 6 types of data filters

- **Scalability:**
  - Collection server handles 1000 + packets/sec
  - 1500 SQL inserts/sec
  - Multicast provides network scalability

- **Usage:**
  - Extensively used with real world application sponsored by DARPA
  - Used by WINLAB projects.
Future work

- **Functionality:**
  - More user friendly API
  - Log application parameter changes with OML

- **OML data filters:**
  - Provision for using 3rd party coded filters
  - Extend the existing library of filters

- **Misc.**
  - Allow finer control of collections framework
  - Load balancing at the collection server
  - Better and user friendly documentation & examples